Tractability and Approximability of Maximal Strip Recovery

Laurent Bulteau, Guillaume Fertin, Minghui Jiang, Irena Rusu

Tractability and Approximability of Maximal Strip Recovery

Laurent Bulteau[1], Guillaume Fertin[1], Minghui Jiang[2], and Irena Rusu[1]

[1] Laboratoire d'Informatique de Nantes-Atlantique (LINA), UMR CNRS 6241
Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3, France
{Laurent.Bulteau,Guillaume.Fertin,Irena.Rusu}@univ-nantes.fr
[2] Department of Computer Science, Utah State University, Logan, UT 84322, USA
mjiang@cc.usu.edu

Abstract. An essential task in comparative genomics is usually to decompose two or more genomes into synteny blocks, that is, segments of chromosomes with similar contents. In this paper, we study the Maximal Strip Recovery problem (MSR) [Zheng et al. 07], which aims at finding an optimal decomposition of a set of genomes into synteny blocks, amidst possible noise and ambiguities. We present a panel of new or improved FPT and approximation algorithms for the MSR problem and its variants. Our main results include the first FPT algorithm for the variant δ-gap-MSR-d, an FPT algorithm for CMSR-d and δ-gap-CMSR-d running in time $O(2.360^k \mathsf{poly}(nd))$, where k is the number of markers or genes considered as erroneous, and a $(d + 1.5)$-approximation algorithm for CMSR-d and δ-gap-CMSR-d.

1 Introduction

An essential task in comparative genomics is usually to decompose two or more genomes into synteny blocks, that is, segments of chromosomes with similar contents. This task is non-trivial when the genomic maps contain noise and ambiguities, which need to be removed before we can give a precise synteny block decomposition. This is the objective of the Maximal Strip Recovery problem (MSR) [10]: to delete a set of markers (genes) from the genomic maps until the remaining markers can be partitioned into a set of strips (synteny blocks) of maximum total length.

We review some definitions. A genome consists of one or more chromosomes; each chromosome is a sequence of genes. Correspondingly, a *genomic map* consists of one or more sequences of gene markers. Each marker is a signed integer representing a gene: the absolute value of the integer represents the family of the gene; the sign of the integer represents the orientation. A marker has *duplicates* if it is contained more than once in some genomic map, possibly in different orientations. A *strip* of $d \geq 2$ genomic maps is a sequence of *at least two* markers appearing consecutively in each map, such that the order of the markers and the orientation of each marker are either both preserved or both reversed. The *reversed opposite* of a sequence $s = \langle x_1, \ldots, x_h \rangle$ is $-s = \langle -x_h, \ldots, -x_1 \rangle$. The

MSR problem on d input maps is the following maximization problem MSR-d [2, 10]:

PROBLEM MSR-d
INPUT: d genomic maps $G_1, \ldots, G_d$ each containing n markers without duplicates.
SOLUTION: d subsequences $G'_1, \ldots, G'_d$ of $G_1, \ldots, G_d$ respectively, each containing the same ℓ markers, such that all the markers in $G'_1, \ldots, G'_d$ can be partitioned into strips.
PARAMETER: the number ℓ of selected markers.

The maximization problem MSR-d that maximizes the parameter ℓ, the number of selected markers, has a *complement* minimization problem called CMSR-d [9, 8] that minimizes the parameter $k = n - \ell$, the number of deleted markers. For genomic maps of close species with few errors, k can be much smaller than ℓ, thus approximation and FPT algorithms are sometimes more relevant for CMSR than for MSR. We refer to Figure 1 for an example.

Given d subsequences $G'_1, \ldots, G'_d$ of d genomic maps $G_1, \ldots, G_d$, respectively, the *gap* between two consecutive markers a and b of G'_i is the number of markers appearing between a and b in G_i, a and b excluded. The *gap* of a strip s is the maximum gap between any two consecutive markers of s in any map G'_i. The deleted markers between markers of a strip correspond to noise and ambiguities, which occur infrequently. A synteny block is a segment of chromosomes that remain undisrupted by genome rearrangements during evolution. Consecutive elements of a synteny block can only be separated in a dataset due to noise and ambiguities. Thus a strip having a large gap is unlikely to correspond to a synteny block; see [3] for an empirical analysis. This leads to the following gap-constrained variant of MSR-d [1]:

PROBLEM δ-gap-MSR-d
INPUT: d genomic maps $G_1, \ldots, G_d$ each containing n markers without duplicates.
SOLUTION: d subsequences $G'_1, \ldots, G'_d$ of $G_1, \ldots, G_d$ respectively, each containing the same ℓ markers, such that all the markers in $G'_1, \ldots, G'_d$ can be partitioned into strips, and such that each strip has gap at most δ.
PARAMETER: the number ℓ of selected markers.

$$
\begin{array}{lrrrrrrrr}
G_1 = & 1 & 5 & -3 & 2 & 6 & 4 & 8 & 7 \\
G_2 = & 1 & 5 & -3 & -8 & 7 & -6 & 2 & 4 \\
G_3 = & -8 & 2 & 7 & -6 & -4 & 3 & -5 & -1
\end{array}
\qquad
\begin{array}{lrrrrr}
G'_1 = & 1 & 5 & -3 & 6 & 8 \\
G'_2 = & 1 & 5 & -3 & -8 & -6 \\
G'_3 = & -8 & -6 & 3 & -5 & -1
\end{array}
$$

Fig. 1. Three genomic maps G_1, G_2, G_3, and an optimal solution G'_1, G'_2, G'_3 for MSR-3. The markers $2, 4, 7$ are deleted; the markers $1, 3, 5, 6, 8$ are selected in two strips $\langle 1, 5, -3 \rangle$ and $\langle 6, 8 \rangle$ of G'_1, G'_2, G'_3. The gap of the strip $\langle 1, 5, -3 \rangle$ is 0; the gap of the strip $\langle 6, 8 \rangle$ is 2, since there are 2 markers between -8 and -6 in G_3.

No doubt that MSR-d is a more elegant problem from a theoretical perspective, but δ-gap-MSR-d could be more relevant in biological applications. The gap-constrained variant of CMSR-d, denoted δ-gap-CMSR-d, can be similarly defined. Similarly to MSR-d and CMSR-d, the parameter for δ-gap-MSR-d is ℓ, and the parameter for δ-gap-CMSR-d is k. In most cases, δ and d are assumed to be constants, although our FPT algorithm in Theorem 3 does not depend on this assumption and can take δ and d as parameters besides ℓ. There is no known direct reduction from δ-gap-MSR-d to MSR-d or vice versa. Although the gap constraint appears to be an additional burden that the algorithm has to take care of, it also limits the set of candidate strips and their intersection pattern, especially when δ is small, which may make the problem easier to handle.

For the four variants of the maximal strip recovery problem, MSR-d, CMSR-d, δ-gap-MSR-d, and δ-gap-CMSR-d, several hardness results have been obtained [2, 9, 6, 1, 7, 8], and a variety of algorithms have been developed, including heuristics [10], approximation algorithms [2, 1, 5], and FPT algorithms [9, 5]. For example, it is known that MSR-d admits a $2d$-approximation algorithm for any $d \geq 2$ [2, 8], and that δ-gap-MSR-d admits a $2d$-approximation algorithm for any $d \geq 2$ and $\delta \geq 1$ and a 1.8-approximation algorithm for $d = 2$ and $\delta = 1$ [1]. Refer also to [11, 5] for some very recent development on the CMSR problem parallel to our work. The following two theorems summarize some basic hardness results regarding these problems:

Theorem 1. [6, 1, 8] MSR-d, CMSR-d, δ-gap-MSR-d, *and* δ-gap-CMSR-d *are* APX*-hard for any* $d \geq 2$ *and* $\delta \geq 2$*, even if all markers appear in positive orientation in all genomic maps;* 1-gap-MSR-d *and* 1-gap-CMSR-d *are NP-hard for any* $d \geq 2$.

Theorem 2. [7] MSR-d *is* W[1]*-hard for any* $d \geq 4$*, even if all markers appear in positive orientation in all genomic maps.*

In this paper, we present a panel of new or improved FPT and approximation algorithms. Our positive results, together with some previous results, are summarized in Table 1. Due to space constraints, we present only three main results in this extended abstract. These results are (i) an FPT algorithm for δ-gap-MSR-d running in time $2^{O(d\delta\ell)}n$, (ii) an FPT algorithm for CMSR-d and δ-gap-CMSR-d, running in time $O(2.360^k \mathsf{poly}(nd))$, and (iii) a $(d+1.5)$-approximation algorithm for CMSR-d and δ-gap-CMSR-d.

Preliminaries. Given a set of genomic maps (either the set of d original maps given as input or some set of reduced maps during the execution of a recursive algorithm), if a maximal sequence of markers form a strip in these maps, then these markers are either all selected or all deleted in *any* optimal solution. This is because any solution that includes only a subset of the markers in a strip can be extended to a better solution that includes all markers in that strip. Hence, these markers can be treated as an atomic unit, and called a *super-marker*, whose *size* is the number of markers it contains. Note that the size of a super-marker is

Problem	Best FPT algorithm (running time)
δ-gap-MSR-d	$O(2^t td\delta^2 + nd\delta)$ [Theorem 3, Section 2] with $t = \ell(1 + \frac{3}{2}d\delta)$
CMSR-d	$O(2.360^k \mathsf{poly}(nd))$ [Theorem 4, Section 3]
δ-gap-CMSR-d ($\delta \geq 2$)	$O(2.360^k \mathsf{poly}(nd))$ [Theorem 4, Section 3]
1-gap-CMSR-d	$O(2^k \mathsf{poly}(nd))$ [See full version]

Problem	Best approximation ratio
MSR-d	$2d$ [2, 8]
δ-gap-MSR-d ($\delta \geq 4$)	$2d$ [1]
1-gap-MSR-d ($d \geq 3$)	$0.75d + 0.75 + \epsilon$ [See full version]
1-gap-MSR-2	1.8 [1]
2-gap-MSR-d	$1.5d + \epsilon$ [See full version]
3-gap-MSR-d	$1.5d + 0.75 + \epsilon$ [See full version]
CMSR-d ($d \geq 3$)	$d + 1.5$ [Theorem 5, Section 4]
CMSR-2	3 [5]
δ-gap-CMSR-d	$d + 1.5$ [Theorem 5, Section 4]
1-gap-CMSR-2	2.778 [See full version]

Table 1. Positive results for variants of MSR.

always at least 2. A marker that does not belong to any super-marker is a *single-marker*. We use the term *single-super-marker* to refer to either a single-marker or a super-marker. A common step of our algorithms is to partition the markers into single-super-markers. If a set of genomic maps contains only super-markers, then we have a straightforward decomposition into strips, without deleting any marker.

For two markers or two single-super-markers u and v, denote by $\mathsf{gap}(u, v)$ the set of markers that appear between u and v in at least one of the maps; clearly $\mathsf{gap}(u, v) = \mathsf{gap}(v, u)$. We call v a *candidate successor* (resp. *candidate predecessor*) of u, and write $v \succ u$ (resp. $v \prec u$), if the following two conditions are satisfied: (1) $\langle u, v\rangle$ (resp. $\langle v, u\rangle$) is a strip (satisfying the δ-gap constraint, if necessary) in the reduced maps with all markers in $\mathsf{gap}(u, v)$ deleted, (2) $\langle u, x, v\rangle$ (resp. $\langle v, x, u\rangle$) cannot be a strip for any $x \in \mathsf{gap}(u, v)$. The relation $\prec$ always refers to two markers of the original map, even if we temporarily work with reduced maps. Note that v is a candidate successor of u if and only if u is a candidate predecessor of v. In the example of Figure 1, we have $6 \prec 8$, and $\mathsf{gap}(6, 8) = \{2, 4, 7\}$. The following lemma gives some basic properties of the function gap:

Lemma 1. (a) *Let u, v, w be three markers or single-super-markers. If u and v are two candidate successors of w with $u \neq v$, then $u \in \mathsf{gap}(w, v)$ and $v \in \mathsf{gap}(w, u)$.* (b) *Let u and v be two single-super-markers. If $u \prec v$ or $u \succ v$, then $\mathsf{gap}(u, v) \neq \emptyset$.*

2 An FPT Algorithm for δ-gap-MSR-d

In this section, we present the first FPT algorithm for δ-gap-MSR-d with the parameter ℓ. Recall that without the gap constraint, MSR-d with the parameter ℓ is W[1]-hard for any $d \geq 4$. In sharp contrast to the W[1]-hardness of MSR-d, we obtain a somewhat surprising result that δ-gap-MSR-d is in FPT, where ℓ is the parameter, and δ and d are constants. In fact, our FPT algorithm for δ-gap-MSR-d works even if d and δ are not constants: δ-gap-MSR-d is in FPT even with three combined parameters d, δ and ℓ.

Theorem 3. *Algorithm 1 finds an optimal solution for* δ-gap-MSR-d *for any* $d \geq 2$ *and* $\delta \geq 1$*, in time* $O(2^t td\delta^2 + nd\delta)$*, where* $t = \ell(1 + \frac{3}{2}d\delta)$.

Algorithm 1 FPT algorithm for δ-gap-MSR-d

1: Gather all pairs of markers (u, v) such that $u \prec v$. Such pairs are called *candidate pairs*.
2: For each marker u, create a boolean variable x_u.
3: For each candidate pair (u, v), create a conjunctive boolean formula $f_{u,v} = x_u \wedge x_v \wedge \neg x_{g_1} \wedge \ldots \wedge \neg x_{g_s}$, where $g_1, \ldots, g_s$ are the markers in $\mathsf{gap}(u, v)$.
4: Delete the variables that do not appear in any formula or appear only in negative form in the formulas.
5: Enumerate all possible assignments to the remaining variables to find an optimal assignment that maximizes the number of variables appearing in positive form in at least one satisfied formula. Delete all markers whose variables are not assigned true values.
6: Return the resulting genomic maps.

Our algorithm is based on a simple idea: create a boolean variable for each marker (where true means the marker is selected in a solution, false that it is unselected), then test all possible assignments to find an optimal solution. To reduce the time complexity of this brute-force approach, we add a pruning step (line 4) to delete certain variables whose markers cannot appear in any optimal solution. The remaining variables form a "kernel" on which we can find an optimal solution in FPT time.

Given an optimal solution, which selects ℓ markers, we call a marker *active* if it appears within distance at most δ from a selected marker in some map. Then each map contains at most $\ell\delta + \frac{\ell}{2}\delta$ unselected active markers: at most δ after each selected marker, and at most δ before the first marker of each strip (note that the number of strips of this optimal solution is at most $\ell/2$). The total number of active markers is at most $\ell + d(\ell\delta + \frac{\ell}{2}\delta) = \ell(1 + \frac{3}{2}d\delta)$.

The pruning step in line 4 depends on the crucial observation that a non-active marker can never appear in positive form. Suppose for contradiction that a non-active marker u appears in a candidate pair with some marker v. Then u is at distance at most $\delta + 1$ from v in each map. Since u, as a non-active marker,

must be at distance at least $\delta + 1$ from the selected markers in all maps, no selected markers can appear between u and v in any map, thus we can extend the optimal solution by selecting both u and v, a contradtiction.

Note that in line 4 the variables appearing at least once in positive form are never deleted, hence no formula becomes empty after deleting the variables that appear only in negative form. After line 4, the number of remaining variables is at most the number of active markers, which is at most $t = \ell(1 + \frac{3}{2}d\delta)$. Correspondingly, the number of formulas is at most $t(\delta+1)$, because any candidate pair consists of an active marker and one of the $\delta + 1$ markers immediately following it in the first map. Each formula contains at most $d\delta + 2$ variables.

The time complexity of line 1 is $O(nd\delta)$. In lines 2 and 3, the variables can be created in time $O(n)$, and the formulas can be created in time $O(t(\delta+1)(d\delta+2)) = O(td\delta^2)$. Similarly, line 4 can be executed in time $O(n + td\delta^2)$. Finally, line 5 can be executed in time $O(2^t t(\delta + 1)(d\delta + 2)) = O(2^t td\delta^2)$, so the overall time complexity is $O(2^t td\delta^2 + nd\delta)$.

3 An FPT Algorithm for CMSR-d and δ-gap-CMSR-d

In this section, we design an FPT algorithm for CMSR-d and δ-gap-CMSR-d, where the parameter is k, the number of deleted markers in the optimal solution.

Since super-markers are already strips in the input genomic maps, one may naturally be tempted to come up with the following algorithm. First, find all super-markers, and add them to the solution. Then, delete a subset of single-markers until all markers in the resulting maps can be partitioned into strips. The correctness of this algorithm on finding an exact solution, however, depends on the assumption that in some optimal solution no super-marker needs to be deleted, which is false as can be seen in the following counter-example:

$$\begin{array}{lccccccc} G_1 = & 4 & 1 & 2 & 3 & 5 & 6 & 7 \\ G_2 = & 6 & -3 & -2 & -1 & 7 & 4 & 5 \end{array}$$

Here $\langle 1, 2, 3\rangle$ forms a super-marker, but the optimal solution deletes $\langle 1, 2, 3\rangle$ and selects $\langle 4, 5\rangle$ and $\langle 6, 7\rangle$ instead. An easy generalization of this counter-example shows that any super-marker of size strictly less than $2d$ is not guaranteed to be always selected in some optimal solution.

We observe that an FPT algorithm for CMSR-d and δ-gap-CMSR-d can be easily obtained using the bounded search tree method. In any feasible solution for the two problems, a single-marker x must be either deleted or selected. If x is selected, then at least one of its neighbors must be deleted. Since x has at most $2d$ neighbors (at most two in each map), this leads to a very simple algorithm running in time $O((2d + 1)^k \mathsf{poly}(nd))$. Parallel to our work, Jiang et al. [5] presented an FPT algorithm running in time $O(3^k \mathsf{poly}(nd))$. We next describe a carefully tuned FPT algorithm running in time $O(2.360^k \mathsf{poly}(nd))$. For convenience, we consider the decision problem associated with CMSR-d and δ-gap-CMSR-d, for which the parameter k is part of the input.

Theorem 4. *Algorithm 2 finds an exact solution for the decision problems associated with* CMSR-d *and* δ-gap-CMSR-d*, for any* $\delta \geq 1$ *and* $d \geq 2$*, in time* $O(c^k \mathsf{poly}(nd))$*, where* $c < 2.360$ *is the unique real root of the equation* $2c^{-1} + 2c^{-3} = 1$*.*

It is interesting to note that although the two problems MSR-d and δ-gap-MSR-d have very different complexities when parameterized by ℓ, their complements CMSR-d and δ-gap-CMSR-d are both tractable when parameterized by k.

Algorithm 2 FPT algorithm for δ-gap-CMSR-d and CMSR-d

Input: d genomic maps $G_1, \ldots, G_d$ each containing n markers without duplicates, and two parameters $k \in \mathbb{N}$, $\delta \in \mathbb{N} \cup \{\infty\}$

return recurse$(G_1, \ldots, G_d, k, \delta,$ false$)$

Function recurse$(G_1, \ldots, G_d, k, \delta,$ skip_step_2b$)$: boolean

if $k < 0$ **then**
  return false
Partition the markers into single-super-markers.
if there exists at least one single-marker in G_1 **then**
  $x \leftarrow$ the left-most single-marker in G_1
else
  return true
$s \leftarrow$ the first single-super-marker following x in G_1
// 1: Assume x is deleted in the optimal solution
Create $G'_1, \ldots, G'_d$ by removing x from $G_1, \ldots, G_d$.
if recurse$(G'_1, \ldots, G'_d, k-1, \delta,$ false$)$ **then**
  return true
// 2: Assume x is part of a strip in the optimal solution
$Y \leftarrow \{$ single-super-marker $y \mid x \prec y\}$ *// the set of candidate successors*
$Z \leftarrow \{$ super-marker $z \mid z \prec x\}$ *// the set of candidate predecessors*
if $\exists w_0 \in Y \cup Z$ a super-marker s.t. (x, w_0) satisfies the conditions of Lemma 2 **then**
  Create $G'_1, \ldots, G'_d$ by removing all markers in gap(x, w_0) from $G_1, \ldots, G_d$.
  return recurse$(G'_1, \ldots, G'_d, k-1, \delta,$ false$)$
if $\exists s_0$ a single-marker s.t. (x, s_0) satisfies the conditions of Lemma 3 **then**
  Create $G'_1, \ldots, G'_d$ by removing s_0 from $G_1, \ldots, G_d$.
  return recurse$(G'_1, \ldots, G'_d, k-1, \delta,$ false$)$
// 2.a: Assume x is not at the end of its strip
if $Y \neq \emptyset$ **then**
  if recurse_2a$(Y, x, G_1, \ldots, G_d, k, \delta)$ **then**
    return true
// 2.b: Assume x is at the end of its strip
if $Z \neq \emptyset$ **and** skip_step_2b=false **then**
  if recurse_2b$(Z, x, s, G_1, \ldots, G_d, k, \delta)$ **then**
    return true
return false

Algorithm 2 (continued)

Function recurse_2a($Y, x, G_1, \ldots, G_d, k, \delta$): boolean

if $\exists y_0 \in Y$ s.t. y_0 satisfies the conditions of Lemma 4 **then**
  if $\delta \in \mathbb{N}$ **and** y_0 is a single-marker **then**
    Replace y_0 by the unspecified marker $[y_0 \mid Y]$.
  $Y_0 \leftarrow \{y_0\}$
else
  $Y_0 \leftarrow Y$
for all $y \in Y_0$ **do**
  Create $G'_1, \ldots, G'_d$ by removing all markers in $\mathsf{gap}(x, y)$ from $G_1, \ldots, G_d$.
  if recurse$(G'_1, \ldots, G'_d, k - |\mathsf{gap}(x, y)|, \delta,$ false$)$ **then**
    return true
return false

Function recurse_2b($Z, x, s, G_1, \ldots, G_d, k, \delta$): boolean

if $\exists z_0 \in Z$ s.t. z_0 satisfies the conditions of Lemma 5 **then**
  $Z_0 \leftarrow \{z_0\}$
else
  $Z_0 \leftarrow Z$
for all $z \in Z_0$ **do**
  if z ends with an unspecified marker $[y_0 \mid Y]$ **and** $\exists y_1 \in Y$ s.t. $y_1 \prec x$ **then**
    Replace the unspecified marker $[y_0 \mid Y]$ by y_1.
  Create $G'_1, \ldots, G'_d$ by removing all markers in $\mathsf{gap}(x, z)$ from $G_1, \ldots, G_d$.
  skip_next_step_2b $\leftarrow$ s exists **and** s is a single-marker **and** $s \notin \mathsf{gap}(x, z)$
  if recurse$(G'_1, \ldots, G'_d, k - |\mathsf{gap}(x, z)|, \delta,$ skip_next_step_2b$)$ **then**
    return true
return false

The efficiency of Algorithm 2 is made possible by several optimizations justified by the following lemmas. These lemmas are all based on very simple observations. Note that although we consider the decision problem for simplicity, Algorithm 2 can be adapted to directly return the actual solution, instead of "true", when the input instance indeed has a solution of size k. Recall that the relation $\prec$ in lines 14-15 is defined for markers in the original maps — it remains unchanged through recursive calls, and can be precomputed.

Lemma 2. *Let x be a single-marker and w a super-marker. If x is selected in an optimal solution, and w is a candidate successor or predecessor of x with exactly one marker in $\mathsf{gap}(x, w)$, then there is an optimal solution where the marker in $\mathsf{gap}(x, w)$ is deleted.*

Lemma 3. *Let x be a single-marker and s a single-super-marker. If s appears in $\mathsf{gap}(x, w)$ for each w that is a candidate successor or predecessor of x, then s itself cannot be a candidate successor or predecessor of x, and any solution selecting x deletes s.*

Lemma 4. *(In this lemma we assume there is no gap constraint.) Let x be a single-marker and y a candidate successor of x such that all markers in $\mathsf{gap}(x, y)$*

are single-markers and candidate successors of x. If x is part of some strip in an optimal solution, but not at the end of this strip, then there is an optimal solution where $\langle x, y\rangle$ is part of some strip.

Lemma 5. *Let x be the first single-marker in G'_1. Let z be a candidate predecessor of x such that all markers in $\mathsf{gap}(x, z)$ are size-2 super-markers and candidate predecessors of x. If x appears at the end of a strip in an optimal solution, then there is an optimal solution where $\langle z, x\rangle$ is at the end of some strip.*

In addition to these four optimizations, we also use a "delayed commitment" optimization which is the equivalent of Lemma 4 when we need to observe a gap constraint. We consider the case where x is part, but not at the end, of some strip in the optimal solution, and where y is a single-marker and a candidate successor of x such that all markers in $\mathsf{gap}(x, y)$ are single-markers and candidate successors of x. In this case we delete all markers in $\mathsf{gap}(x, y)$ to make $\langle x, y\rangle$ a strip, but keep the possibility of replacing y by any marker $y_1 \in \mathsf{gap}(x, y)$, should necessity arise. We denote this unspecified marker by $[y \mid \mathsf{gap}(x, y)]$.

To prove the correctness of Algorithm 2, we need the following easy lemma from [10]:

Lemma 6. [10, Proposition 2] *We can decompose the strips of any optimal solution in such a way that (1) each strip contains at most 3 single-super-markers and (2) each strip containing 3 single-super-markers starts and ends with a single-marker.*

Let OPT be any optimal solution, and let us decompose the strips of OPT as in the above lemma. We show by induction that the solution found by Algorithm 2 has the same size as OPT. Let x be the left-most single-marker in G_1, then exactly one of the following three cases is true:

1: x is deleted in OPT,
2.a: There exists a single-super-marker y such that $\langle x, y\rangle$ is part of a strip in OPT,
2.b: There exists a super-marker z such that $\langle z, x\rangle$ is a strip in OPT.

Note that in case 2.b, z cannot be a single-marker since it is to the left of x in G_1. By our choice of x, case 2.a can be split into the following two subcases:

2.a.i: There exists a single-super-marker y such that $\langle x, y\rangle$ is a strip in OPT,
2.a.ii: There exists a single-super-marker y and a single-marker y' such that $\langle x, y, y'\rangle$ is a strip in OPT.

Refer to Algorithm 2. In case 1, a solution is found in lines 9–12 of the function recurse. In case 2, i.e. in the case where x is part of an optimal solution, if either Lemma 2 or Lemma 3 can be applied, then again a solution is found. Otherwise, we are in case 2.a or 2.b.

Suppose we are in case 2.a. If $y \in Y_0$, then the function recurse_2a tests a branch in which $\langle x, y\rangle$ becomes part of some strip. Otherwise, there exists some

Algorithm 3 $(d + 1.5)$-approximation for δ-gap-CMSR-d and CMSR-d

```
1: X ← { triples of markers (z, x, y) | z ≺ y and gap(z, y) = {x} }
2: Partition the markers into single-super-markers.
3: for all (z, x, y) ∈ X do
4:    if x, y and z are not deleted and y or z is a single-marker then
5:       Delete x.
6:       Re-create all super-markers.
7: Delete all remaining single-markers.
8: Return the resulting genomic maps.
```

$y_0 \in Y$ satisfying the conditions of Lemma 4. If there is no gap constraint, y is replaced by y_0, which does not change the size of the solution. If there is a gap constraint, y is replaced by the unspecified marker $u = [y_0 \mid Y]$, and we look further in case 2.a.i or 2.a.ii.

In case 2.a.i, we can replace y by y_0 since $\mathsf{gap}(x, y_0)$ has no more markers than $\mathsf{gap}(x, y)$. In case 2.a.ii, we can replace y by any y_1 such that $x \prec y_1 \prec y'$, since $\mathsf{gap}(x, y) \cup \{y\} \cup \mathsf{gap}(y, y')$ is the same set as $\mathsf{gap}(x, y_1) \cup \{y_1\} \cup \mathsf{gap}(y_1, y')$. This is what happens in case 2.b of a subsequent recursive call in which y' becomes the left-most single-marker in G_1.

Suppose we are in case 2.b. If $z \in Z_0$, then the function recurse_2b tests a branch in which $\langle z, x \rangle$ becomes a strip. Otherwise, Lemma 5 can be applied, which leaves the size of the optimal solution unchanged. In line 9 of recurse_2b, if s becomes the left-most single-marker in G_1 in the next recursive call of recurse, it cannot be at the end of a strip because x is already at the end of a strip.

This completes the correctness proof. An anonymous reviewer of an earlier version of this paper commented that perhaps some further properties of the optimal solution, besides those already described in our lemmas, might be used to improve the time complexity further. This may be true, but we believe that such improvement would require significantly different ideas.

4 An Approximation Algorithm for CMSR-d and δ-gap-CMSR-d

In this section, we present a $(d + 1.5)$-approximation algorithm for the two minimization problems CMSR-d and δ-gap-CMSR-d. Recall that $2d$-approximation algorithms [2, 8, 1] were known for the two maximization problems MSR-d and δ-gap-MSR-d.

Theorem 5. *Algorithm 3 finds a* $(d + 1.5)$*-approximation for* CMSR-d *and* δ-gap-CMSR-d *for any* $d \geq 2$ *and* $\delta \geq 1$.

Let k be the number of deleted markers in an optimal solution. Then the number of single-markers in the input maps is at most $(2d + 1)k$ because each single-marker is either deleted or adjacent to a deleted marker. This immediately yields a $(2d + 1)$-approximation algorithm: simply delete all single-markers.

$$
\begin{array}{lccccc}
G_1 = z_d y_d & \cdots & z_3 y_3 & z_2 y_2 & z_1\, x\, y_1 \\
G_2 = z_1 y_1 & z_2\, x\, y_2 & z_3 y_3 & \cdots & z_d y_d \\
G_3 = z_1 y_1 & z_2 y_2 & z_3\, x\, y_3 & \cdots & z_d y_d \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
G_d = z_1 y_1 & z_2 y_2 & z_3 y_3 & \cdots & z_d\, x\, y_d
\end{array}
$$

Fig. 2. A tight example for the $(2d+1)$-approximation algorithm. The optimal solution deletes one single-marker x instead of all $2d+1$ single-markers.

We refer to Figure 2 for a tight example for the $(2d+1)$-approximation algorithm. Observe that after one single-marker is deleted, many other single-markers may be merged into strips. Algorithm 3 first identifies (line 1) all triples of markers (z, x, y) such that z and y can be merged into a strip $\langle z, y\rangle$ after x is deleted, then successively deletes (lines 2–6) "cost-efficient" single-markers x that can reduce at least one other single-marker y or z, and finally removes (line 7) the remaining single-markers.

Lemma 7. *For each triple (z, x, y) in the set X in Algorithm 3, at least one of the three markers x, y, z must be deleted in any feasible solution.*

Proof. We prove the lemma by contradiction. Suppose that all three markers x, y, z are selected in a solution. Assume wlog that the sequence $\langle z, x, y\rangle$ appears in some map. Then x must be in the same strip as z or y. Assume wlog that $\langle z, x\rangle$ is part of some strip. Then $z \prec x$. Recall that $z \prec y$. Thus x and y are both candidate successors of z. By Lemma 1a, we have $y \in \mathsf{gap}(z, x)$, thus y must be deleted: a contradiction. □

We next prove the approximation ratio of Algorithm 3. Let O be the set of deleted markers in an optimal solution; $|O| = k$. For each marker $x \notin O$, we define two sets $\mathit{\Gamma}_{succ}(x)$ and $\mathit{\Gamma}_{pred}(x)$ as follows. If x is followed by a marker y in a strip of O, $\mathit{\Gamma}_{succ}(x) = \mathsf{gap}(x, y)$; otherwise x is the last marker of its strip, $\mathit{\Gamma}_{succ}(x) = \emptyset$. If x is preceded by a marker z in a strip of O, $\mathit{\Gamma}_{pred}(x) = \mathsf{gap}(z, x)$; otherwise x is the first marker of its strip, $\mathit{\Gamma}_{pred}(x) = \emptyset$. Then, for each marker $x \notin O$, define $\gamma(x) = |\mathit{\Gamma}_{succ}(x)| + |\mathit{\Gamma}_{pred}(x)|$, and for each marker $x \in O$, define $\gamma(x) = 0$.

Refer to Algorithm 3. Let D be the set of markers deleted in line 5, let S be the set of single-markers that are merged into super-markers in line 6, and let R be the set of markers deleted in line 7. Let $R_1 = \{r \in R \mid \gamma(r) = 1\}$ and $R_2 = \{r \in R \mid \gamma(r) \geq 2\}$. Note that if x is a single-marker at the beginning of the algorithm, then $\gamma(x) = 0$ if and only if $x \in O$. Thus we have a partition $R = (R \cap O) \cup R_1 \cup R_2$. Also note that each marker $x \in O$ is counted by γ at most twice in each map: at most once in some $\mathit{\Gamma}_{pred}(y)$, and at most once in some $\mathit{\Gamma}_{succ}(z)$. Thus we have the following inequality:

$$\sum_{x \text{ single-marker}} \gamma(x) \leq 2dk. \tag{1}$$

Each marker $x \in D$ has a corresponding triple $(z, x, y) \in X$, where z or y is a single-marker. After x is deleted in line 5, z and y are merged into the same super-marker in line 6. Thus we have the following inequality:

$$|D| \leq |S|. \tag{2}$$

For each marker $x \in D - O$, let $\phi(x)$ be an arbitrary marker in the non-empty set $\{z, x, y\} \cap O$ (see Lemma 7). Obviously $\phi(x) \neq x$, thus $\phi(x) \in O - D$. We show that at most two markers in $D - O$ can have the same image by ϕ. Suppose that $\phi(x_1) = \phi(x_2) = \phi$ for two different markers $x_1, x_2 \in D - O$, where x_1 is deleted before x_2 in Algorithm 3. Then the marker ϕ is merged into a super-marker after x_1 is deleted, and again merged into a larger super-marker after x_2 is deleted. Since a marker has at most two neighbors in a super-marker, ϕ is necessarily a single-marker before x_1 is deleted, so it belongs to S, indeed $S \cap O$. Moreover, after x_2 is deleted and ϕ is merged into a larger super-marker, ϕ cannot be adjacent to any other single-marker, say x_3. Therefore

$$|D - O| \leq |O - D| + |S \cap O|. \tag{3}$$

Let u be a marker such that $\gamma(u) = 1$. Then u belongs to some strip in the optimal solution, and it has a neighbor $v = \psi(u)$ in the same strip such that $\mathsf{gap}(u, v)$ contains only one marker, say x. Note that $u, v \notin O$ and $x \in O$. We claim that if u is a single-marker at the beginning of the algorithm, then either $u \in D \cup S$ or $v \in D$. This claim is clearly true if u or v is deleted by the algorithm in line 5. Otherwise, with $(v, x, u) \in X$ or $(u, x, v) \in X$, either x is not deleted because u is merged into a super-marker, or x is deleted: in both cases $u \in S$. This proves the claim. So for each $u \in R_1$, we have $v \in D$, indeed $v \in D - O$. Note that there can be at most two markers u_1 and u_2 with the same image v by ψ: the two neighbors of v in some strip in the optimal solution. Thus we have $|R_1| \leq 2|D - O|$. Moreover, if there are two markers u_1 and u_2 with the same image v, then $\gamma(v) \geq 2$. Therefore

$$|R_1| \leq \sum_{v \in D - O} \gamma(v). \tag{4}$$

Combining inequalities (1), (2), (3), and (4), the calculation in the following shows that the number of deleted markers, $|D| + |R|$, is at most $(d + 1.5)k$. Thus Algorithm 3 indeed finds a $(d+1.5)$-approximation for δ-gap-CMSR-d and CMSR-d.

$$\begin{aligned}
2dk &\geq \sum_{x \text{ single-marker}} \gamma(x) \quad \text{by (1)} \\
&= \sum_{x \in D - O} \gamma(x) + \sum_{x \in S - O} \gamma(x) + \sum_{x \in R_1} \gamma(x) + \sum_{x \in R_2} \gamma(x) \\
&\geq \sum_{x \in D - O} \gamma(x) + |S - O| + |R_1| + 2|R_2| \\
&\geq |S - O| + 2|R_1| + 2|R_2| \quad \text{by (4)}.
\end{aligned}$$

$$
\begin{aligned}
|D| + |R| &= |D| + |R_1| + |R_2| + |R \cap O| \\
&\le |D| + dk - \tfrac{1}{2}|S - O| + |R \cap O| \\
&= |D| + dk - \tfrac{1}{2}(|S| - |S \cap O|) + |R \cap O| \\
&\le |D| + dk - \tfrac{1}{2}|D| + \tfrac{1}{2}|S \cap O| + |R \cap O| \quad \text{by (2)} \\
&= \tfrac{1}{2}(|D| + |S \cap O|) + |R \cap O| + dk \\
&= \tfrac{1}{2}(|D \cap O| + |D - O| + |S \cap O|) + |R \cap O| + dk \\
&\le \tfrac{1}{2}(|D \cap O| + (|O - D| + |S \cap O|) + |S \cap O|) + |R \cap O| + dk \quad \text{by (3)} \\
&= \tfrac{1}{2}|O| + (|S \cap O| + |R \cap O|) + dk \\
&\le \tfrac{1}{2}k + k + dk \\
&= \left(d + \tfrac{3}{2}\right) k.
\end{aligned}
$$

After our initial submission of this paper for publication, we learned from an anonymous reviewer that Jiang et al. [5] have very recently designed a 3-approximation algorithm for CMSR-2 based on a similar greedy approach. Their algorithm does not work for the gap-constrained variant, although it seems that the algorithm might be extended to a $(d+1)$-approximation for CMSR-d for all $d \ge 2$. Our solution gives uniform results on both variants.

$$
\begin{array}{lccccccc}
G_1 = & z_d y_d & \cdots & z_3 y_3 & z_2 y_2 & z_1\, vu\, y_1 \\
G_2 = & z_1 y_1 & z_2\, uv\, y_2 & z_3 y_3 & \cdots & z_d y_d \\
G_3 = & z_1 y_1 & z_2 y_2 & z_3\, uv\, y_3 & \cdots & z_d y_d \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
G_d = & z_1 y_1 & z_2 y_2 & z_3 y_3 & \cdots & z_d\, uv\, y_d
\end{array}
$$

$$
\begin{array}{lccccccc}
G_1 = & z_d y_d & \cdots & z_3 y_3 & z_2 y_2 & z_1\, -v-u\, y_1 \\
G_2 = & z_1 y_1 & z_2\, uv\, y_2 & z_3 y_3 & \cdots & z_d y_d \\
G_3 = & z_1 y_1 & z_2 y_2 & z_3\, uv\, y_3 & \cdots & z_d y_d \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
G_d = & z_1 y_1 & z_2 y_2 & z_3 y_3 & \cdots & z_d\, uv\, y_d
\end{array}
$$

Fig. 3. Upper: an almost-tight example for the $(d+1.5)$-approximation algorithm showing that its approximation ratio cannot be better than $d+1$; the optimal solution deletes the two single-markers u and v instead of all $2d+2$ single-markers. Lower: an example showing that no algorithm deleting only single-markers can achieve an approximation ratio better than d; the optimal solution deletes one super-marker $\langle u, v \rangle$ instead of $2d$ single-markers z_i and y_i, $1 \le i \le d$.

We refer to Figure 3 for two examples: the first example gives a lower bound of $d+1$ on the approximation ratio of Algorithm 3; the second example gives a lower bound of d on the approximation ratio of *any* algorithm for δ-gap-CMSR-d and CMSR-d that deletes only single-markers. Note that both our Algorithm 3 and the algorithm in [5] delete only single-markers.

Compared to the approximation upper bound of $2d$ [2, 1, 8] for the two maximization problems MSR-d and δ-gap-MSR-d, which almost matches (at least

asymptotically) the current best lower bound of $\Omega(d/\log d)$ [8], our upper bound of $d+1.5$ for the two minimization problems CMSR-d and δ-gap-CMSR-d is still far away from the constant lower bound in [8]. It is an intriguing question whether CMSR-d and δ-gap-CMSR-d admit approximation algorithms with constant ratios independent of d.

References

1. L. Bulteau, G. Fertin, and I. Rusu. Maximal strip recovery problem with gaps: hardness and approximation algorithms. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC'09)*, LNCS 5878, pages 710–719, 2009.
2. Z. Chen, B. Fu, M. Jiang, and B. Zhu. On recovering syntenic blocks from comparative maps. *Journal of Combinatorial Optimization*, 18:307–318, 2009.
3. V. Choi, C. Zheng, Q. Zhu, and D. Sankoff. Algorithms for the extraction of syntheny blocks from comparative maps. In *Proceedings of the 7th International Workshop on Algorithms in Bioinformatics (WABI'07)*, LNCS 4645, pages 277–288, 2007.
4. M. M. Halldórsson. Approximating discrete collections via local improvements. In *Proceedings of the 6th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'95)*, pages 160–169, 1995.
5. H. Jiang, Z. Li, G. Lin, L. Wang, and B. Zhu. Exact and approximation algorithms for the complementary maximal strip recovery problem. *Journal of Combinatorial Optimization*, doi:10.1007/s10878-010-9366-y, to appear.
6. M. Jiang. Inapproximability of maximal strip recovery. In *Proceedings of the 20th International Symposium on Algorithms and Computation (ISAAC'09)*, LNCS 5878, pages 616–625, 2009.
7. M. Jiang. On the parameterized complexity of some optimization problems related to multiple-interval graphs. *Theoretical Computer Science*, to appear. A preliminary version in *Proceedings of the 21st Annual Symposium on Combinatorial Pattern Matching (CPM'10)*, LNCS 6129, pages 125–137, 2010.
8. M. Jiang. Inapproximability of maximal strip recovery: II. In *Proceedings of the 4th International Frontiers of Algorithmics Workshop (FAW'10)*, LNCS 6213, pages 53–64, 2010.
9. L. Wang and B. Zhu. On the tractability of maximal strip recovery. In *Proceedings of the 6th Annual Conference on Theory and Applications of Models of Computation (TAMC'09)*, LNCS 5532, pages 400–409, 2009.
10. C. Zheng, Q. Zhu, and D. Sankoff. Removing noise and ambiguities from comparative maps in rearrangement analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 4:515–522, 2007.
11. B. Zhu. Efficient exact and approximate algorithms for the complement of maximal strip recovery. In *Proceedings of the 6th International Conference on Algorithmic Aspects in Information and Management (AAIM'10)*, LNCS 6124, pages 325-333, 2010.